# OSProj1 Compile Linux Kernel & Introduction to Linux Kernel Modules

by 潘禧辰，518021910497

## Menu

## Abstract

- 将Linux内核编译为了最新版本的5.5.8
- 创建了simple模块
- 创建了jiffies模块，获取jiffies值
- 创建了seconds模块，获取seconds值

## Environment

- Ubuntu 18.04
- Linux 5.3.0-42-generic
- VMware Workstation Rro 15.5.0 build-14665864

## Quick Start

Linux内核编译在下文中做阐述。

这里对于另外三个模块的测试方法进行说明：

### Makefile的编写

以jiffies模块为例，编写如下所示的Makefile文件，对于其他两个模块只需要修改object name即可

```
obj-m := jiffies.o
all:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules
clean:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

### 测试代码

以jiffies模块为例，测试时使用如下代码，其他的模块对应修改模块名即可，其中simple模块不需要进行因为没有`proc_read`函数不需要进行`cat`操作。

```
sudo dmesg -c                #定期清除缓存区
make                         #编译
sudo insmod jiffies.ko       #加载jiffies内核模块
dmesg                        #显示proc_init函数中要打印的东西，即The /proc/jiffies
loaded!
cat /proc/jiffies            #显示proc_read函数中要打印的东西
sudo rmmod jiffies           #删除jiffies内核模块
dmesg                        #显示proc_exit函数中要打印的东西，检测是否正常删除
```

## Background

- Linux是开源的操作系统，内核版本不断更新，新的内核修订了旧内核的bug，并增加了许多新的特性。而Ubuntu系统中的内核往往不是最新发布的，如果用户想要使用这些新特性，或想根据自己的系统度身定制一个更高效，更稳定的内核，就需要重新编译内核。
- Linux 内核模块作为 Linux 内核的扩展手段，可以在运行时动态加载和卸载。它是设备和用户应用程序之间的桥梁，可以通过标准系统调用，为应用程序屏蔽设备细节，本次project就是在内核态下进行编程。

## Implementation & Result

### Compile Linux Kernel

#### 安装VMware

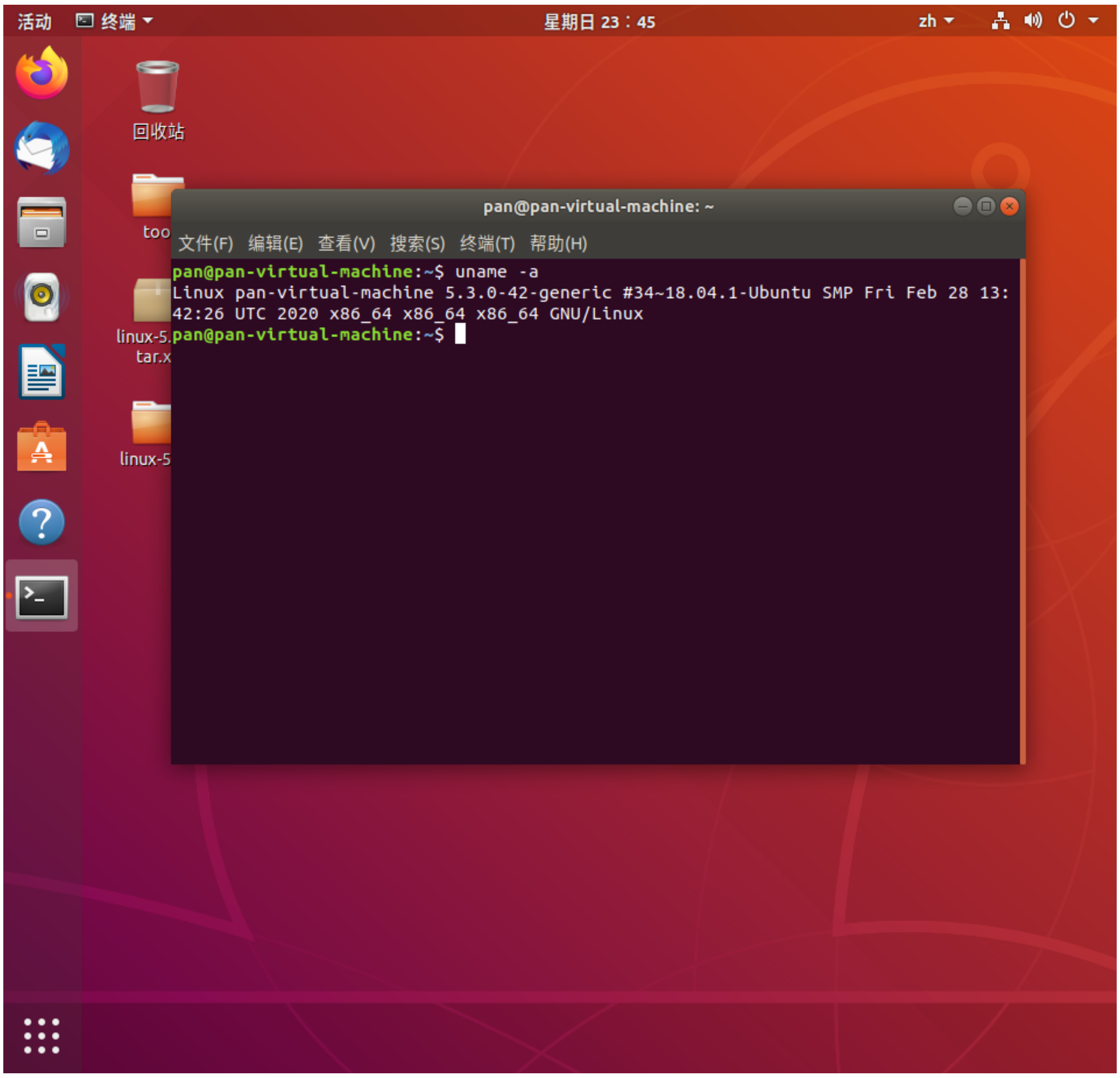
使用上海交通大学授权进行安装，此处不赘述

#### 安装Ubuntu 18.04

- 到Ubuntu官网https://ubuntu.com/download/desktop下载镜像
- 按照VMware引导进行内存和核心分配
- 按照Ubuntu引导进行系统分区
- 按照VMware引导安装tool

#### 编译Linux 5.5.8内核

- 在Windows中到官网https://www.kernel.org/下载Linux 5.5.8内核，并使用tool拖拽至虚拟机中
- 使用`uname -a`查看当前内核版本

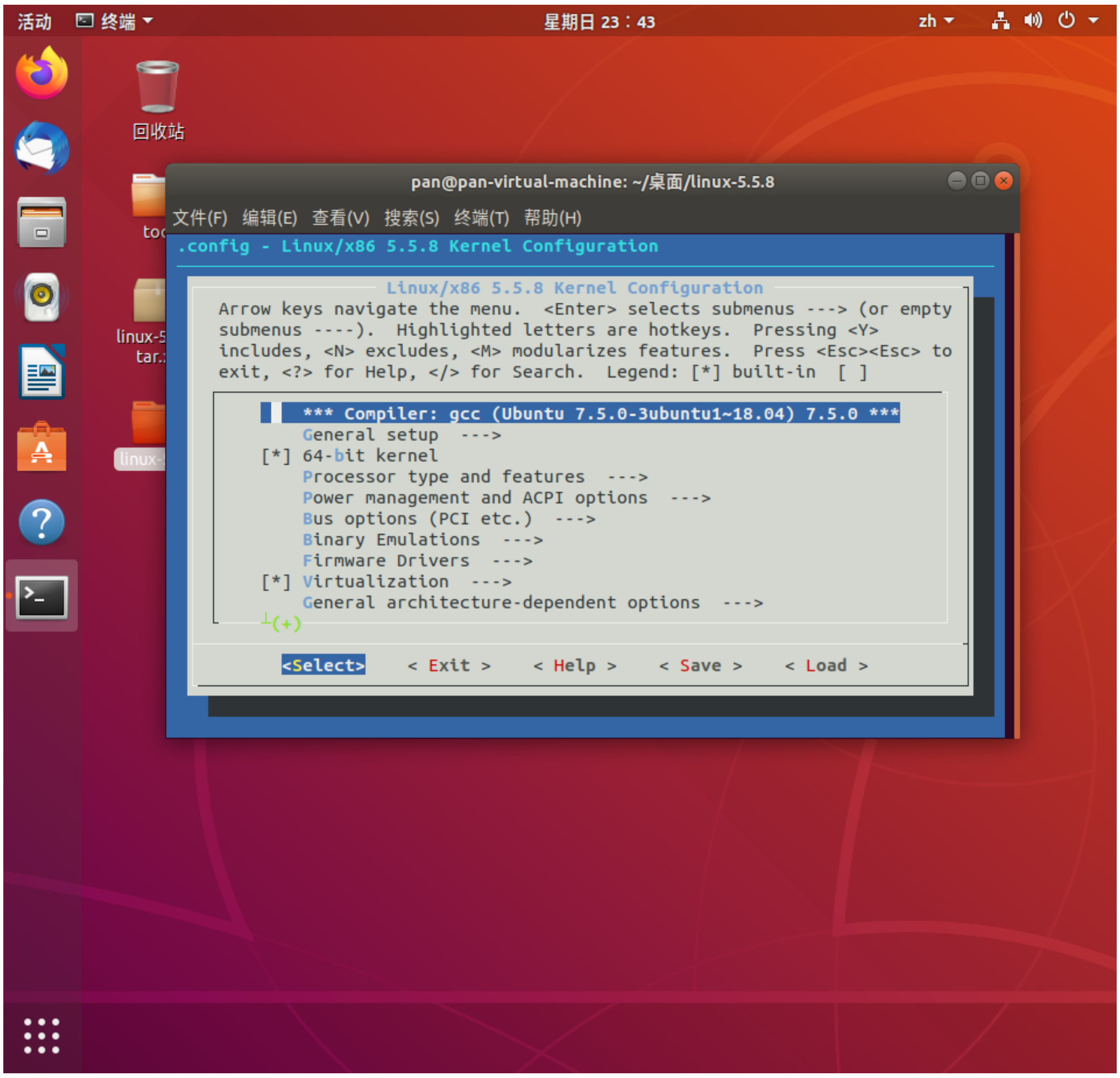

活动
终端
星期日 23：45
zh
回收站
too
linux-5.
tar.x
linux-5
pan@pan-virtual-machine: ~
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

```
pan@pan-virtual-machine:~$ uname -a
Linux pan-virtual-machine 5.3.0-42-generic #34~18.04.1-Ubuntu SMP Fri Feb 28 13:
42:26 UTC 2020 x86_64 x86_64 x86_64 GNU/Linux
pan@pan-virtual-machine:~$
```

- 安装必要依赖库

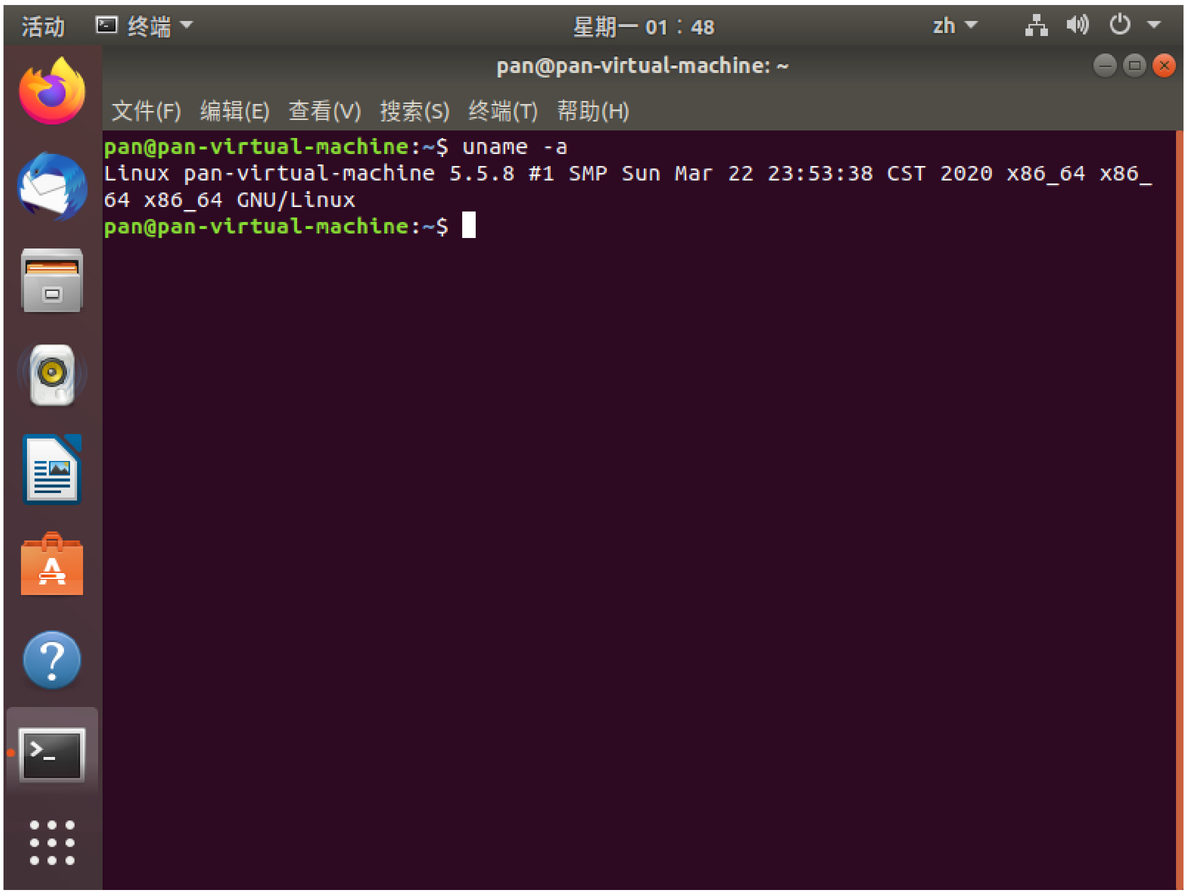
```
sudo apt-get install gcc make libncurses5-dev openssl libssl-dev
sudo apt-get install build-essential
sudo apt-get install pkg-config
sudo apt-get install libc6-dev
sudo apt-get install bison
sudo apt-get install flex
sudo apt-get install libelf-dev
```

- 使用指令`tar -zxf linux-5.5.8.tar.xz`解压内核
- 使用`sudo make menuconfig`指令在GUI下调整内核config，本次使用默认config


活动
终端
星期日 23：43
zh
回收站
pan@pan-virtual-machine: ~/桌面/linux-5.5.8
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

```
.config - Linux/x86 5.5.8 Kernel Configuration
                    Linux/x86 5.5.8 Kernel Configuration
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty
  submenus ----).  Highlighted letters are hotkeys.  Pressing <Y>
  includes, <N> excludes, <M> modularizes features.  Press <Esc><Esc> to
  exit, <?> for Help, </> for Search.  Legend: [*] built-in  [ ]

        *** Compiler: gcc (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0 ***
            General setup  --->
        [*] 64-bit kernel
            Processor type and features  --->
            Power management and ACPI options  --->
            Bus options (PCI etc.)  --->
            Binary Emulations  --->
            Firmware Drivers  --->
        [*] Virtualization  --->
            General architecture-dependent options  --->
        ┴(+)

          <Select>    < Exit >    < Help >    < Save >    < Load >
```

- 使用指令`sudo make`开始进行编译
- 使用指令`make modules_install`安装编译后的各个模块
- 使用指令`make install`完成安装
- 使用`uname -a`查看编译后内核版本


活动
终端
星期一 01：48
zh
pan@pan-virtual-machine: ~
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

```
pan@pan-virtual-machine:~$ uname -a
Linux pan-virtual-machine 5.5.8 #1 SMP Sun Mar 22 23:53:38 CST 2020 x86_64 x86_
64 x86_64 GNU/Linux
pan@pan-virtual-machine:~$
```

## Introduction to Linux Kernel Modules

### simple

simple模块的编写主要参考了Operating System Concept书中的介绍。在书中原有的基础上，要求附加4个功能：

1. Print out the value of `GOLDEN_RATIO_PRIME` in the `simple_init()` function.
2. Print out the greatest common divisor of 3,300 and 24 in the `simple_exit()` function.
3. Print out the values of jiffies and HZ in the `simple_init()` function.
4. Print out the value of jiffies in the `simple_exit()` function.

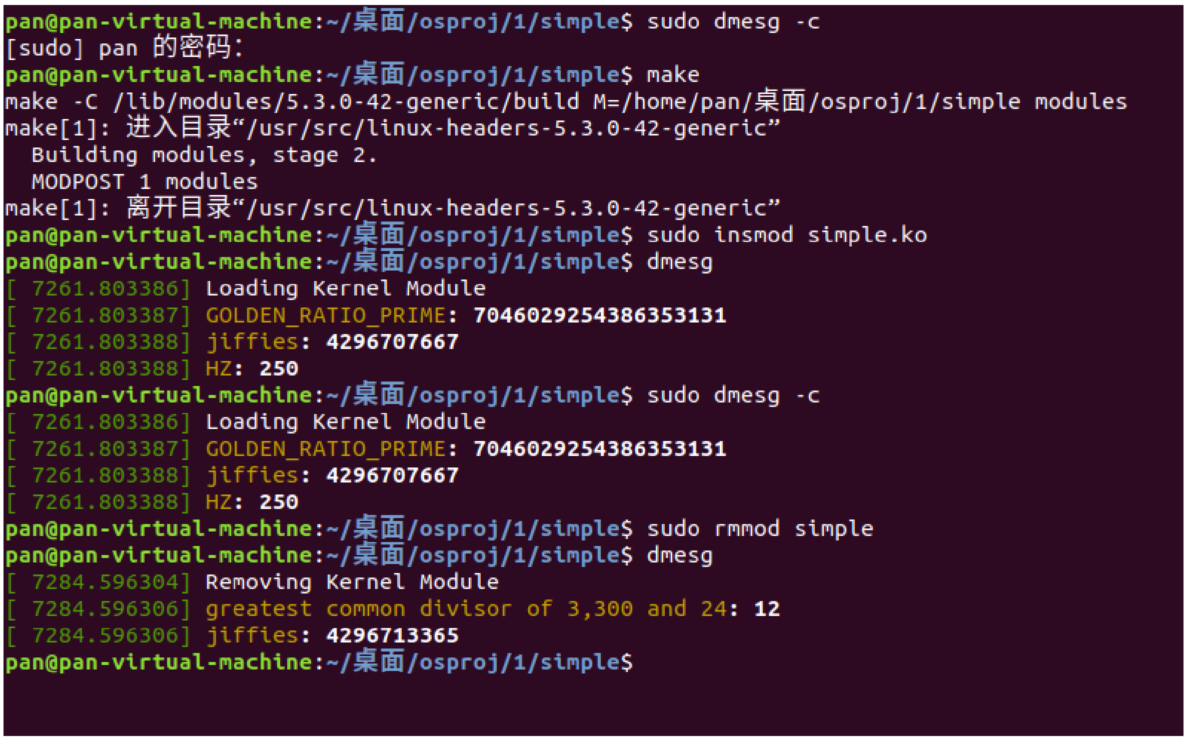
代码如下：

```
#include <linux/init.h>
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/hash.h>
#include <linux/gcd.h>
#include <linux/jiffies.h>

/* This function is called when the module is loaded. */
int simple_init(void)
{
    printk(KERN_INFO "Loading Kernel Module\n");
    printk("GOLDEN_RATIO_PRIME: %llu\n", GOLDEN_RATIO_PRIME);
    printk("jiffies: %lu\n", jiffies);
    printk("HZ: %u\n", HZ);
    return 0;
}

/* This function is called when the module is removed. */
```

```
void simple_exit(void)
{
    printk(KERN_INFO "Removing Kernel Module\n");
    printk("greatest common divisor of 3,300 and 24: %lu\n", gcd(3300,24));
    printk("jiffies: %lu\n", jiffies);
}

/* Macros for registering module entry and exit points. */
module_init(simple_init);
module_exit(simple_exit);

MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("proj1-1");
MODULE_AUTHOR("Xichen Pan");
```

在原始代码基础上，修改了`simple_init(void)`和`simple_exit(void)`两个函数，添加了一些要求的输出内容。

结果如下：



```
pan@pan-virtual-machine:~/桌面/osproj/1/simple$ sudo dmesg -c
[sudo] pan 的密码：
pan@pan-virtual-machine:~/桌面/osproj/1/simple$ make
make -C /lib/modules/5.3.0-42-generic/build M=/home/pan/桌面/osproj/1/simple modules
make[1]: 进入目录“/usr/src/linux-headers-5.3.0-42-generic”
  Building modules, stage 2.
  MODPOST 1 modules
make[1]: 离开目录“/usr/src/linux-headers-5.3.0-42-generic”
pan@pan-virtual-machine:~/桌面/osproj/1/simple$ sudo insmod simple.ko
pan@pan-virtual-machine:~/桌面/osproj/1/simple$ dmesg
[ 7261.803386] Loading Kernel Module
[ 7261.803387] GOLDEN_RATIO_PRIME: 7046029254386353131
[ 7261.803388] jiffies: 4296707667
[ 7261.803388] HZ: 250
pan@pan-virtual-machine:~/桌面/osproj/1/simple$ sudo dmesg -c
[ 7261.803386] Loading Kernel Module
[ 7261.803387] GOLDEN_RATIO_PRIME: 7046029254386353131
[ 7261.803388] jiffies: 4296707667
[ 7261.803388] HZ: 250
pan@pan-virtual-machine:~/桌面/osproj/1/simple$ sudo rmmod simple
pan@pan-virtual-machine:~/桌面/osproj/1/simple$ dmesg
[ 7284.596304] Removing Kernel Module
[ 7284.596306] greatest common divisor of 3,300 and 24: 12
[ 7284.596306] jiffies: 4296713365
pan@pan-virtual-machine:~/桌面/osproj/1/simple$
```

**jiffies**

jiffies模块要求：

Design a kernel module that creates a /proc file named `/proc/jiffies` that reports the current value of `jiffies` when the `/proc/jiffies` file is read, such as with the command `cat /proc/jiffies`

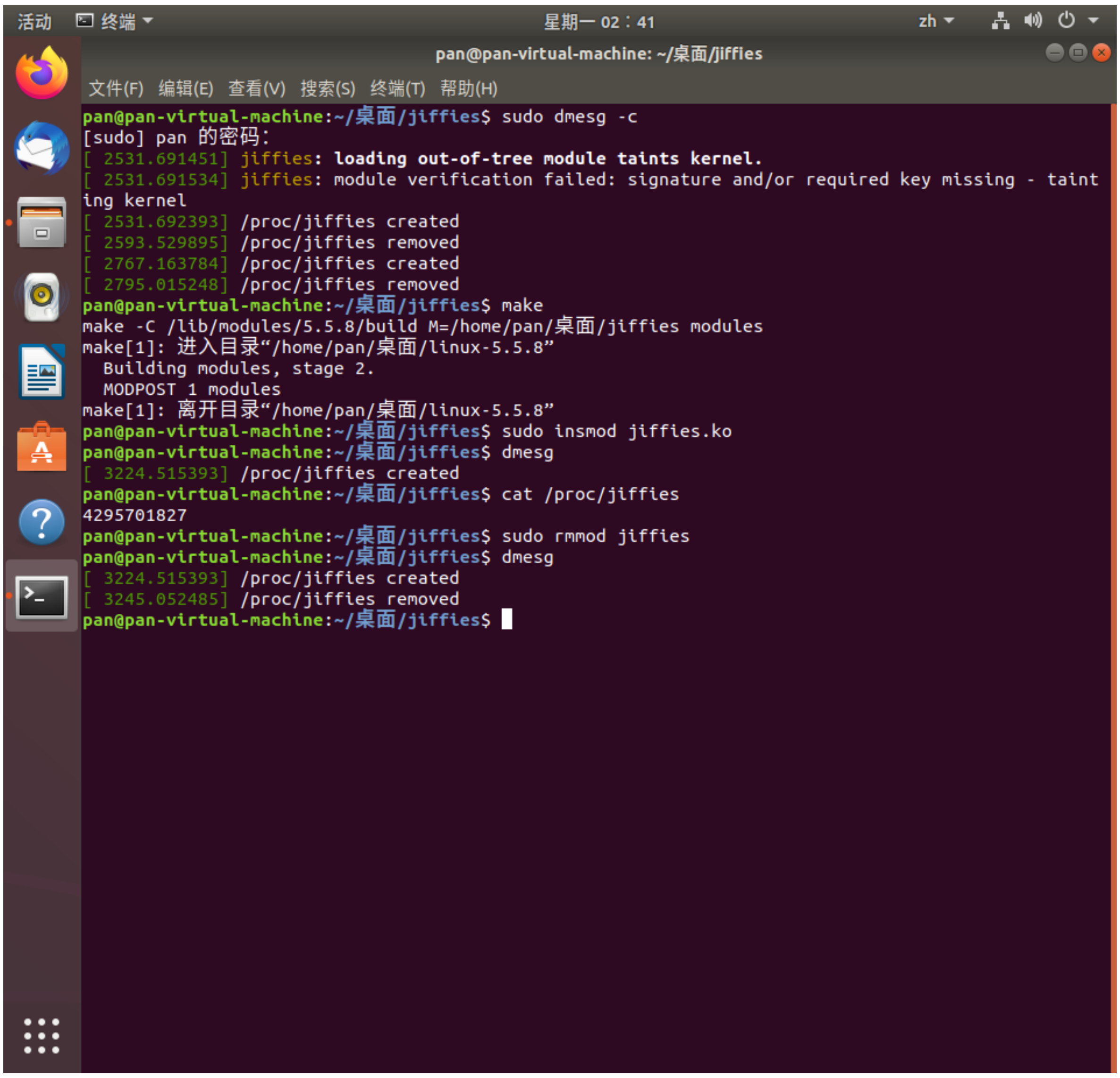
代码如下：

```
#include <linux/init.h>
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/proc_fs.h>
#include <linux/uaccess.h>
#include <linux/jiffies.h>
```

```c
#define BUFFER_SIZE 128
#define PROC_NAME "jiffies"

ssize_t proc_read(struct file *file, char __user *buf, size_t count, loff_t
*pos);

static struct file_operations proc_ops = {
    .owner = THIS_MODULE,
    .read = proc_read,
};

/* This function is called when the module is loaded. */
int proc_init(void) {
    /* creates the /proc/hello entry */
    proc_create(PROC_NAME, 0, NULL, &proc_ops);
    printk(KERN_INFO "/proc/%s created\n", PROC_NAME);
    return 0;
}

/* This function is called when the module is removed. */
void proc_exit(void) {
    /* removes the /proc/hello entry */
    remove_proc_entry(PROC_NAME, NULL);
    printk(KERN_INFO "/proc/%s removed\n", PROC_NAME);
}

ssize_t proc_read(struct file *file, char __user *usr_buf, size_t count, loff_t
*pos)
{
    int rv = 0;
    char buffer[BUFFER_SIZE];
    static int completed = 0;

    if (completed) {
        completed = 0;
        return 0;
    }

    rv = sprintf(buffer, "%lu\n", jiffies);

    /* copies kernel space buffer to user space usr buf */
    copy_to_user(usr_buf, buffer, rv);

    completed = 1;
    return rv;
}
module_init( proc_init );
module_exit( proc_exit );

MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("proj1-2");
MODULE_AUTHOR("Xichen Pan");
```

在原始代码基础上，修改了`proc_read`函数，添加了指令`rv = sprintf(buffer, "%lu\n", jiffies);`，在`buffer`中输出了`jiffies`的值。

结果如下：


活动
终端
星期一 02：41
zh
pan@pan-virtual-machine: ~/桌面/jiffies
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

```
pan@pan-virtual-machine:~/桌面/jiffies$ sudo dmesg -c
[sudo] pan 的密码：
[ 2531.691451] jiffies: loading out-of-tree module taints kernel.
[ 2531.691534] jiffies: module verification failed: signature and/or required key missing - taint
ing kernel
[ 2531.692393] /proc/jiffies created
[ 2593.529895] /proc/jiffies removed
[ 2767.163784] /proc/jiffies created
[ 2795.015248] /proc/jiffies removed
pan@pan-virtual-machine:~/桌面/jiffies$ make
make -C /lib/modules/5.5.8/build M=/home/pan/桌面/jiffies modules
make[1]: 进入目录"/home/pan/桌面/linux-5.5.8"
  Building modules, stage 2.
  MODPOST 1 modules
make[1]: 离开目录"/home/pan/桌面/linux-5.5.8"
pan@pan-virtual-machine:~/桌面/jiffies$ sudo insmod jiffies.ko
pan@pan-virtual-machine:~/桌面/jiffies$ dmesg
[ 3224.515393] /proc/jiffies created
pan@pan-virtual-machine:~/桌面/jiffies$ cat /proc/jiffies
4295701827
pan@pan-virtual-machine:~/桌面/jiffies$ sudo rmmod jiffies
pan@pan-virtual-machine:~/桌面/jiffies$ dmesg
[ 3224.515393] /proc/jiffies created
[ 3245.052485] /proc/jiffies removed
pan@pan-virtual-machine:~/桌面/jiffies$ 
```

### seconds

seconds模块要求：

Design a kernel module that creates a proc file named `/proc/seconds` that reports the number of `elapsed seconds` since the kernel module was
loaded. This will involve using the value of `jiffies` as well as the `HZ` rate. When a user enters the command `cat /proc/seconds`

因为有：$elapsed\ seconds = \frac{jiffies_1 - jiffies_0}{HZ}$，所以只需要在`proc_init()`中记录初始`jiffies`值到`t0`中，然后在`proc_read`函数中使用公式`(jiffies - t0) / HZ`计算输出`elapsed seconds`即可。

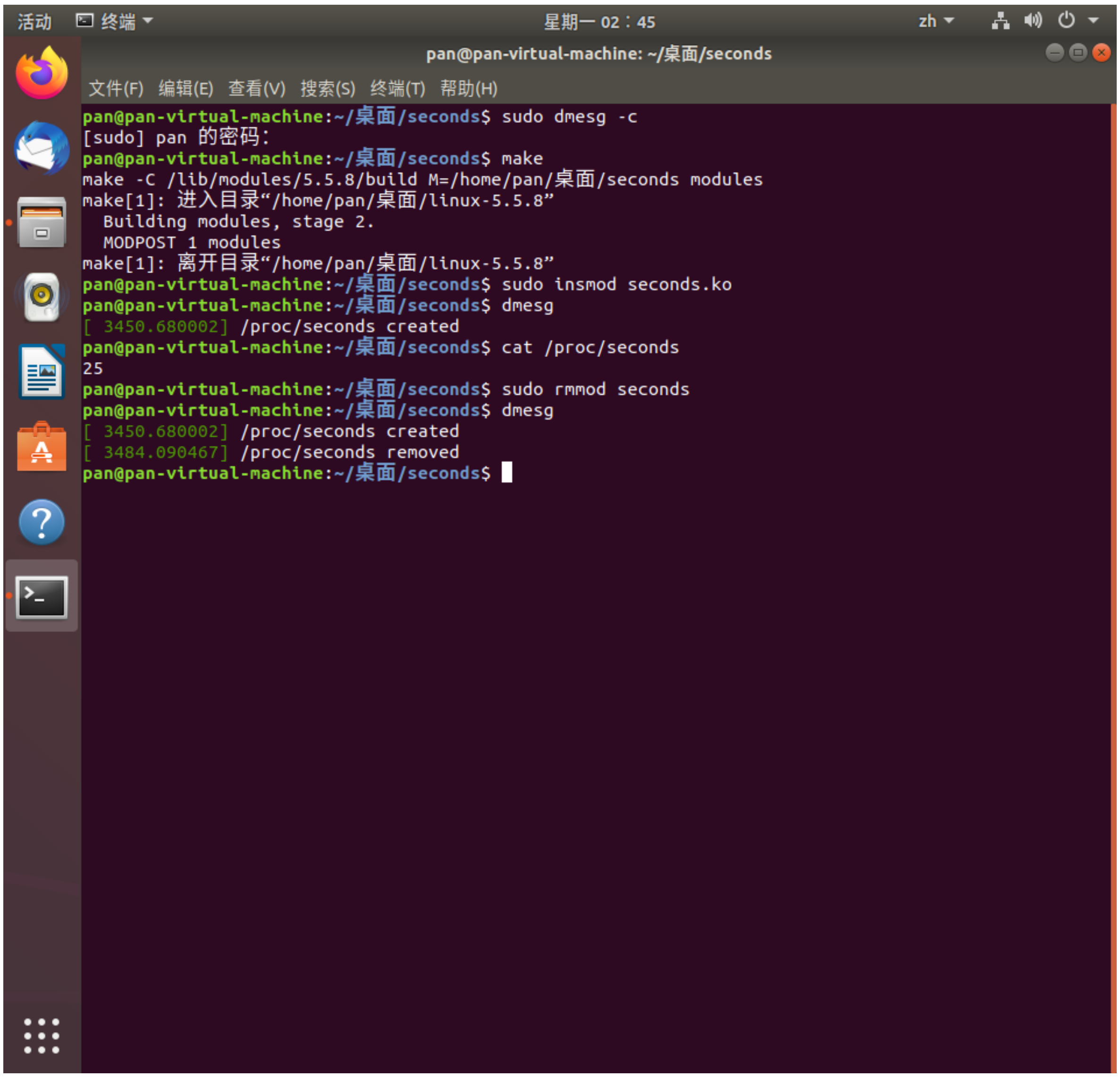
代码如下：

```
#include <linux/init.h>
#include <linux/kernel.h>
#include <linux/module.h>
#include <linux/proc_fs.h>
#include <linux/uaccess.h>
#include <linux/jiffies.h>

#define BUFFER_SIZE 128
#define PROC_NAME "seconds"

unsigned long t0 = 0;
```

```
ssize_t proc_read(struct file *file, char __user *buf, size_t count, loff_t
*pos);

static struct file_operations proc_ops = {
    .owner = THIS_MODULE,
    .read = proc_read,
};

/* This function is called when the module is loaded. */
int proc_init(void)
{
    /* creates the /proc/hello entry */
    proc_create(PROC_NAME, 0, NULL, &proc_ops);
    printk(KERN_INFO "/proc/%s created\n", PROC_NAME);
    t0 = jiffies;
    return 0;
}

/* This function is called when the module is removed. */
void proc_exit(void)
{
    /* removes the /proc/hello entry */
    remove_proc_entry(PROC_NAME, NULL);
    printk( KERN_INFO "/proc/%s removed\n", PROC_NAME);
}

/* This function is called each time /proc/hello is read */
ssize_t proc_read(struct file *file, char __user *usr_buf, size_t count, loff_t
*pos)
{
    int rv = 0;
    char buffer[BUFFER_SIZE];
    static int completed = 0;

    if (completed) {
        completed = 0;
        return 0;
    }

    rv = sprintf(buffer, "%lu\n", (jiffies - t0) / HZ);

    /* copies kernel space buffer to user space usr buf */
    copy_to_user(usr_buf, buffer, rv);

    completed = 1;
    return rv;
}
module_init( proc_init );
module_exit( proc_exit );

MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("proj1-3");
MODULE_AUTHOR("Xichen Pan");
```

在原始代码基础上，修改了`proc_init`函数和`proc_read`函数，在`buffer`中输出了`elapsed seconds`的值。

结果如下：


活动
终端
星期一 02：45
zh
pan@pan-virtual-machine: ~/桌面/seconds
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)

```
pan@pan-virtual-machine:~/桌面/seconds$ sudo dmesg -c
[sudo] pan 的密码：
pan@pan-virtual-machine:~/桌面/seconds$ make
make -C /lib/modules/5.5.8/build M=/home/pan/桌面/seconds modules
make[1]: 进入目录“/home/pan/桌面/linux-5.5.8”
  Building modules, stage 2.
  MODPOST 1 modules
make[1]: 离开目录“/home/pan/桌面/linux-5.5.8”
pan@pan-virtual-machine:~/桌面/seconds$ sudo insmod seconds.ko
pan@pan-virtual-machine:~/桌面/seconds$ dmesg
[ 3450.680002] /proc/seconds created
pan@pan-virtual-machine:~/桌面/seconds$ cat /proc/seconds
25
pan@pan-virtual-machine:~/桌面/seconds$ sudo rmmod seconds
pan@pan-virtual-machine:~/桌面/seconds$ dmesg
[ 3450.680002] /proc/seconds created
[ 3484.090467] /proc/seconds removed
pan@pan-virtual-machine:~/桌面/seconds$
```

## Difficulties

- 因为是第一次编写Makefile文件，而且Operating System Concept书中也没有对Makefile的编写进行详尽的介绍，花费了不少时间学习Makefile的语法。
- 第一次project总体比较简单，内核编译也一遍通过了，没有遇到太大的困难。

## Reference

- Operating System Concept $10^{th}$ edition
- VMware安装Ubuntu18.04 https://zhuanlan.zhihu.com/p/38797088
- Vmware+Ubantu 编译linux内核 https://blog.csdn.net/zhangkai9895/article/details/104700914